IEEE Std 1609.2-2016 Guidance Note 5: Root CA permissions

William Whyte, Security Innovation

Version 0.4, 2016-09-09

1 Introduction

This note identifies an area in IEEE Std 1609.2-2016 that has been reported as implemented differently to the spec in a significant deployed implementation. The note also proposes a modified interpretation of the spec that provides minimal disruption to deployed implementations.

The defect concerns the statement of permissions in the root CA certificate. The root CA certificate contains permissions that allow it to issue, directly or via another intermediate CA, certificates that can be used to sign SCMS-internal management messages (SMMs), misbehavior reports (MBR), or certificate revocation lists (CRLs). However, due to an incorrect interpretation of the spec when the certificate profile of the root CA certificate was written, the root CA can in fact only authorize the issuance of a limited subset of SMM-, MBR- or CRL-signing certificates: it was intended to be able to authorize *all SSPs* for those application areas (a Service Specific Permission, or SSP, is a field used to authorize specific activities within an overall application area such as “issuing CRLs” or “signing SMMs”), but, if a strict interpretation of 1609.2 is followed, the root CA cert as issued can only authorize the *default SSP* for the application area.

This means that under the current interpretation of 1609.2, and with the current root CA certificate, **there is no entity in the system capable of authorizing**, for example, **a Registration Authority (RA) to send SCMS-internal messages**; there is also **no entity in the system capable of authorizing anyone to sign any CRL**.

The specific mechanism by which this issue arose is as follows.

In 1609.2 permissions, application areas are identified by Provider Service Identifiers (PSIDs); specific activities within those application areas are identified by Service Specific Permissions (SSPs).

An *end-entity certificate* (a certificate that signs application messages) uses a structure called a PsidSsp to indicate exactly what messages it can sign (or, more broadly, what application activities it can carry out). The PsidSsp structure contains a PSID and a single SSP.

A CA certificate uses a structure called a PsidSspRange to indicate the permissions that the CA may include in certificates it issues. The PsidSspRange structure contains a single PSID and EITHER no SSP at all, OR a list of SSPs, OR an indication that the CA can issue all SSPs associated

with that PSID. This allows a single CA certificate to issue different end-entity certificates that contain different PsidSsp values, so long as each of them are consistent with the CA's PsidSspRange.[1]

The problem arises in the interpretation of the PsidSspRange when the SSP Range is omitted.

PsidSspRange is defined in 1609.2 clause 6.4.33 as follows:

```
 PsidSspRange ::= SEQUENCE {
    psid                Psid,
    sspRange            SspRange OPTIONAL
}
```

SspRange is defined in 1609.2 clause 6.4.34 as follows:

```
SspRange ::= CHOICE {
    opaque              SequenceOfOctetString,
    all                 NULL,
    ...
}
```

The different options for SspRange, per 1609.2 clause 6.4.33, are:

- If the SspRange indicates the option *opaque*, it is a list of SSPs and the CA may authorize any SSP on the list.
- If the SspRange indicates the option *all*, the CA may authorize any SSP associated with the PSID.
- If the SspRange is *omitted*, the CA may authorize *only the default SSP* associated with the PSID.

The Root CA certificate **should have its SspRange for MBR, SMM and CRL indicate the option all, but instead the SspRange was omitted for those PSIDs**. This means that the Root CA certificate can only issue the default SSP for MBR, SMM and CRL, and there is no entity in the system that is authorized to issue any other SSP for those application areas.

The relevant part of the Root CA certificate profile is below. Note that sspRange is ABSENT for each of the last three PsidGroupPermissions.

[1] The CA may have more than one PsidSspRange in its certificate, but that doesn't affect this explanation.

```
certIssuePermissions (SequenceOfPsidGroupPermissions (SIZE(4)) (CONSTRAINED BY {
  PsidGroupPermissions ( WITH COMPONENTS {...,
        subjectPermissions (WITH COMPONENTS {
       all
    }),
    minChainDepth(3),
    chainDepthRange(-1),
    eeType ({app, enrol})
  }),
  PsidGroupPermissions ( WITH COMPONENTS {...,
        subjectPermissions (WITH COMPONENTS{
      explicit (SequenceOfPsidSspRange (SIZE (1)) (WITH COMPONENT (WITH COMPONENTS {
        psid (SecurityMgmtPsid), sspRange ABSENT
      }))))
    }),
    minChainDepth(1),
    chainDepthRange(-1),
        eeType ({app, enrol})
  }),
  PsidGroupPermissions ( WITH COMPONENTS {...,
        subjectPermissions (WITH COMPONENTS{
      explicit (SequenceOfPsidSspRange (SIZE (1)) (WITH COMPONENT (WITH COMPONENTS {
        psid (MisbehaviorReportingPsid), sspRange ABSENT
      }))))
    }),
    minChainDepth(1),
    chainDepthRange(-1),
        eeType ({app, enrol})
  }),
  PsidGroupPermissions ( WITH COMPONENTS {...,
        subjectPermissions (WITH COMPONENTS{
      explicit (SequenceOfPsidSspRange (SIZE (1)) (WITH COMPONENT (WITH COMPONENTS {
        psid (CrlPsid), sspRange ABSENT
      }))))
    }),
    minChainDepth(1),
    chainDepthRange(-1),
        eeType ({app, enrol})
  })
})),
```

2 Impact

A correct implementation of 1609.2 will reject all MBR, SMM and CRL signing certificates as invalid.

This is particularly impactful in two cases:

- CRLs are received by end-entity devices. End-entity devices will reject CRLs because they do not have a valid chain of issuance, because the root CA was not entitled to authorize that specific CRL signer.
- All MBR certificates will be considered invalid. However, the certificates that end devices use to sign their MBRs are also the certificates that are used to sign BSMs. If a certificate is invalid for one use, it is invalid for any use. Therefore, **any device that strictly checks the validity of the chain will consider all incoming BSMs invalid, and any device that strictly checks the validity of the chain will consider its own BSM signing certificates to be invalid and will not send any BSMs.**

3 Possible solutions

Two possible solutions have been identified.

1. Reissue the root certificate with permissions that are correct per 1609.2-2016. This would mean that all certificates that have been issued to date would need to be withdrawn and reissued. At this stage this is a significant number of certificates.

2. Issue instructions that 1609.2 be interpreted differently such that:
    - If the SspRange is *omitted*, the CA may authorize any SSP associated with the PSID.
    - If the SspRange contains only a single zero-length octet string, the CA may authorize *only the default SSP* associated with the PSID.

Option 1 would be high-impact and would penalize the innovators who have taken on the risk of making early deployment work.

Option 2 is a change to the standard and could in principle lead to inconsistent implementations. However, we do not know of any implementations that currently implement certificate permission checking in full. Also, we do not know of any implementations that fail to interoperate with the issued root CA certificate. The impact of this is therefore considerably lower.

4 Recommendation

We recommend that 1609.2 is modified as follows.

6.4.33 PsidSspRange

```
PsidSspRange ::= SEQUENCE {
    psid                Psid,
    sspRange            SspRange OPTIONAL
}

SequenceOfPsidSspRange ::= SEQUENCE OF PsidSspRange
```

This structure represents the certificate issuing or requesting permissions of the certificate holder with respect to one particular set of application permissions. In this structure:

— `psid` identifies the application area.

— `sspRange` identifies the SSPs associated with `psid` for which the holder may issue or request certificates. If sspRange is omitted, the holder may issue or request certificates for any SSP for `psid`.

6.4.34 SspRange

```
SspRange ::= CHOICE {
    opaque              SequenceOfOctetString,
```

```
    all                     NULL,
    ...
}
```

This structure identifies the SSPs associated with a PSID for which the holder may issue or request certificates.

— If the choice indiated is `opaque`, the certificate holder may issue or request certificates with the listed SSPs for that PSID.

— If the choice indicated is `all`, the holder may issue or request certificates for any SSP for that PSID.

NOTE 1: The choice "`all`" may also be indicated by omitting the SspRange in the enclosing PsidSspRange structure. Omitting the SspRange is preferred to explicitly indicating "`all`"

NOTE 2: Permissions to issue or request the default PSID are indicated by choosing the type "`opaque`" and including a zero-length OCTET STRING within the `SequenceOfOctetString` structure.